Implementing Networking Using the Cocoa Classes

Session 132

These are confidential sessions—please refrain from filming or taking pictures.

Application Technologies

Implementing Networking Using the Cocoa Classes

Chris Parker
Cocoa Frameworks Engineer

https://developer.apple.com/wwdcsecure/

What You'll Learn

- Aspects of networking in Cocoa
- How to write basic TCP servers and clients
- How to use Bonjour effectively
- When to use what classes
  - CFSocket
  - NSStream
  - NSURL
  - CFHTTPMessage
- Several examples

Basic Clients and Servers

http://developer.apple.com/wwdc2005/

TCPServer

- Uses CFSocket to listen for incoming connections
- Publishes a port on both IPv4 and IPv6
- Advertises the service with Bonjour
- Delegate-based connection handling
- Subclassable

CocoaEcho

- Provides a simple echo service
- Server
  - Subclasses TCPServer
  - Manages connection pairs
- Client
  - GUI application
  - Browses for services

TCPServer and CocoaEcho

- What they do
  - Offer both IPv4 and IPv6 services
  - Publish with Bonjour
  - Handle events asynchronously
- What they don’t do
  - Treat IPv4 and IPv6 differently
  - Show the user hostnames or ports
  - Resolve the discovered services

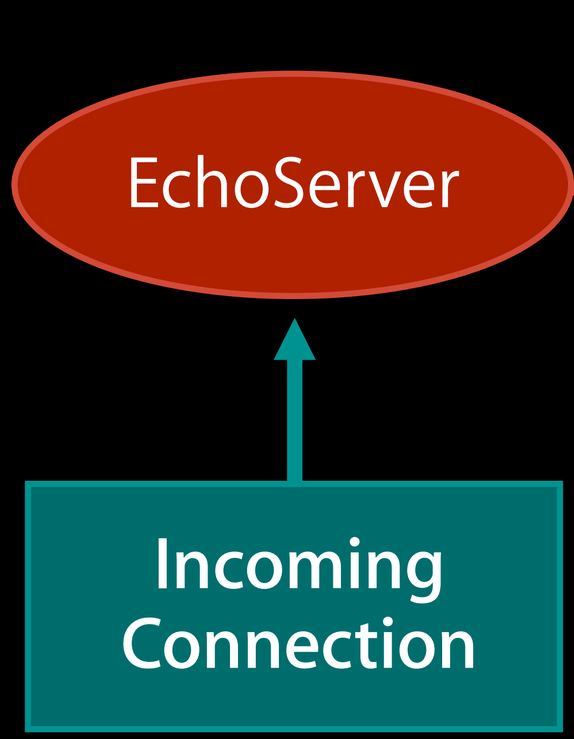
CocoaEcho

Classes

CocoaEcho	EchoServer, CocoaEchoClient
Foundation	NSInputStream, NSOutputStream, NSNetServices
CFNetwork	
CoreFoundation	CFSocket
BSD Sockets	getpeername, setsockopt

EchoServer

Handling new incoming connections



Connections Dictionary

Key	Value
NSInputStream	NSOutputStream

EchoServer is the delegate of the new NSInputStream and NSOutputStream

EchoServer

Handling data on NSInputStreams

Code and Demo

TCPServer and CocoaEcho

http://developer.apple.com/wwdc2005/

TCPServer and CocoaEcho

- What they do
  - Offer both IPv4 and IPv6 services
  - Publish with Bonjour
  - Handle events asynchronously
- What they don’t do
  - Treat IPv4 and IPv6 differently
  - Show the user hostnames or ports
  - Resolve the discovered services

HTTPServer

- Subclasses TCPServer
- Produces HTTPConnection objects for new connections
- Inherits other TCPServer behavior
  - Published with Bonjour
  - IPv4 and IPv6 services

HTTPConnection

- Produces HTTPRequest objects and informs the delegate
- Sends responses in the order requests were received
  - Not necessarily the order the delegate fulfills the requests
- Basic GET method processing if no delegate is present
- Shows more sophisticated asynchronous network event handling

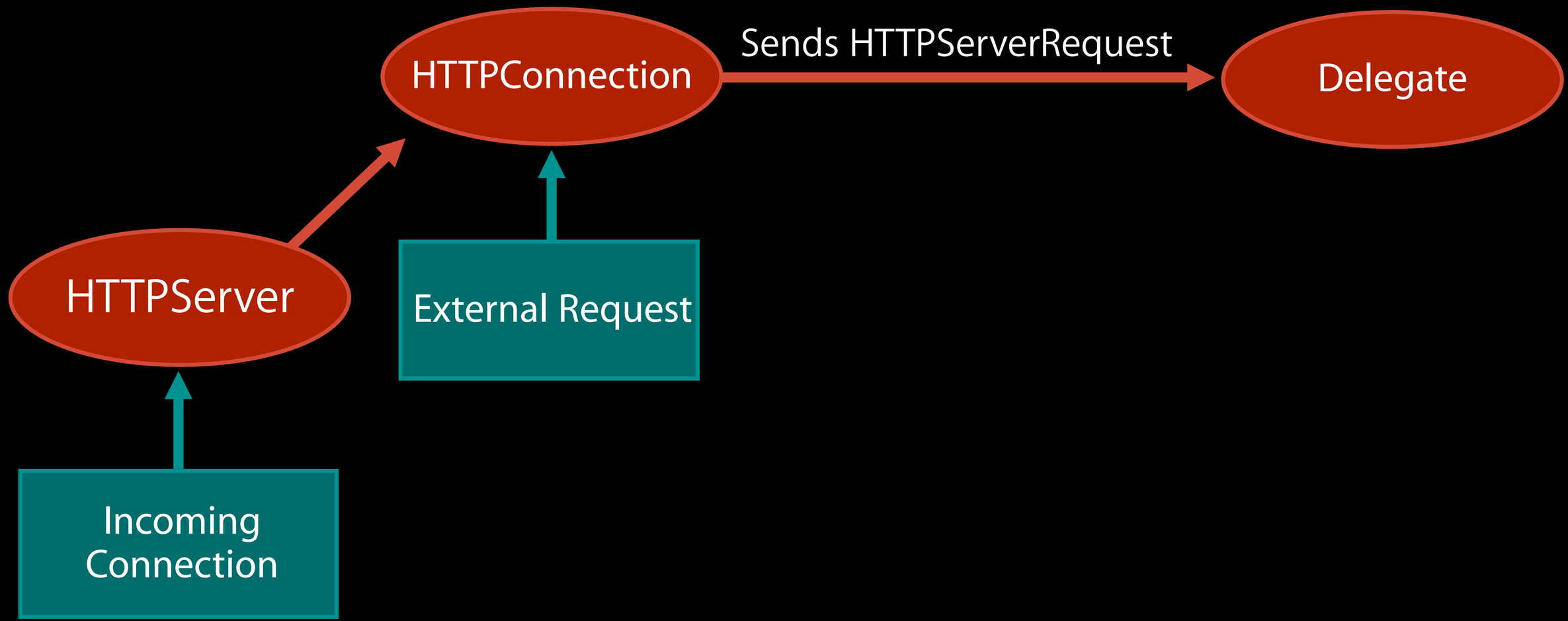
HTTPServer

Classes

HTTPServer	HTTPServer, HTTPConnection, HTTPServerResponse
Foundation	NSURL
CFNetwork	CFHTTPMessage
CoreFoundation	
BSD Sockets	

HTTPServer

Handling connection events



Code and Demo

HTTPServer

http://developer.apple.com/wwdc2005/

HTTPServer

- Not a complete HTTP server
  - Defers most of the actual HTTP protocol handling to the delegate
  - Missing “chunked” transfer coding support
- RFC 2616 describes HTTP 1.1

Application Update

Douglas Davidson
Cocoa Frameworks Engineer

http://developer.apple.com/wwdc2005/

AutoUpdater

- Demo application does one thing
  - Check for and download updated versions of itself
- Most of the work is in determining and implementing specific policies the developer wants
  - When does the application check for updates?
  - Can version X on system Y be updated to version Z?
  - Is the user authorized to obtain an update?
- We leave that up to you
- Hint: put as little policy in the application as possible

What Do You Need?

- One web address to hard-code into your application
  - There needs to be a server running at this address “forever”
- Server, either HTTP server or something else
- Basic set of policies to implement initially
- Version info in the application, license info if necessary
- User interface in the application

Which Classes to Use?

- Want high-level interface for client-side use of web protocols
- Solution: NSURL loading classes
- Used by WebKit for loading and downloading of resources
  - Deal with web as a browser would
  - By default will handle http:, https:, ftp:, and file: protocols
  - Supersede NSURLHandle
- Principal NSURL loading classes:
  - NSURLRequest
  - NSURLResponse
  - NSURLConnection
  - NSURLDownload

NSURL Loading Classes

- NSURL represents arbitrary URL
- NSURLRequest represents load request using arbitrary protocol
  - URL, caching policy, timeout
  - For HTTP: request method, header fields, request body
  - NSMutableURLRequest lets you modify any of these
- NSURLResponse represents response metadata
  - URL, MIME type, expected content length, text encoding
  - For HTTP: status code, header fields
  - Can potentially come from network or from cache

NSURL Loading Classes (Cont.)

- NSURLConnection handles loading into memory
- NSURLDownload handles downloading to a file
  - Allows for resumption, decoding, suggested filename, etc.
- Response data is delivered asynchronously, messaging the delegate
  - NSURLConnection delivers data to delegate
  - NSURLDownload transfers data to file, tells delegate about it
- In either case, delegate is informed of events including
  - Redirects
  - Authentication challenges
  - Response
  - Receipt of data
  - Success or failure

Advanced Topics

- Detailed cache management
- Cookie handling
- Authentication
- Adding other protocols beyond the built-in http:, https:, ftp: and file:

Update Process

- When time is right, application checks for update
- Application sends its version and license info to server
  - Example includes them in URL of a GET request
  - Alternatives include POST, SOAP, custom protocol
- Server responds with information on available updates
- Application presents this to the user
- User confirms whether they want to update or not
- New version is downloaded
  - Example uses a disk image
  - Alternatives include package, updater, and patches
- Security best handled end-to-end

Application Update

Classes

App. Update	Controller
Foundation	NSURLConnection, NSURLDownload
CFNetwork	
CoreFoundation	
BSD Sockets	

Code and Demo

Application Update

http://developer.apple.com/wwdc2005/

Application Update Summary

- Use NSURL loading classes for high-level web client interface
- Use NSURLConnection for loading into memory
- Use NSURLDownload for downloading to a file
- Implement delegate to handle their asynchronous messages

SOAP

Chris Kane
Cocoa **Frameworks** Engineer

http://developer.apple.com/wwdc2005/

SOAP

- W3C standard
- Framework for exchanging messages on the Internet
- Current version of spec is 1.2 (2003)
- SOAP can be used for one- or two-way messaging
- Access to many web services is via SOAP
  - Example: google.com
- SOAP could be used as transport layer for Distributed Objects
- Use WebServicesCore API in CoreServices

SOAP as a Framework

- Very open and flexible
- SOAP over HTTP via POST requests is common
- SOAP-compliant messaging has few constraints
  - Data must be structured as an XML Infoset
  - Does not require XML 1.0 format
  - Does not require any particular transmission protocol
  - Does not prescribe any data typing standard
- Clients must provide information in the way the service wants
- Example will simply cover SOAP messaging with Objective C
  - Will reuse HTTPServer for server side

SOAP
Classes

SOAP	SOAPAdder
Foundation	NSXMLDocument, NSURLConnection
CFNetwork	
CoreFoundation	
BSD Sockets	

Code and Demo

SOAP

http://developer.apple.com/wwdc2005/

SOAP

- Example could be generalized, abstracted further
- But, customizing hooks are then needed
  - And the hooks have to be set up…
- Underlying technologies eliminate most work
- Or, of course, use WebServicesCore

Conclusion

http://developer.apple.com/wwdc2005/

Conclusion

- TCPServer, Echo Server
  - Wrapped CFSockets, publishing via Bonjour, streams on the socket
- HTTPServer
  - Used CFHTTPMessage
- AutoUpdater
  - Used NSURLRequest, NSURLConnection
- SOAP
  - Used NSURLConnection, NSXMLDocument (new in Tiger!)

Q&A

Matthew Formica
Cocoa Evangelist
mformica@apple.com

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference